Core Java :
Date Time API(java.time)

Persistent Interactive | Persistent University

Objectives :

At the end of this module, you will be able to understand:

- Why new API for date & time?
- Core Ideas of new Design
- Listing of classes and interfaces in new API
- Creating Date and time instances
- Formatting Dates

Why New API?

- Date & SimpleDateFormatter are not threadSafe
- Poor API Design list date starts with *1900-1-0* which is start with zero and doesn't looks natural

Core Ideas of new Design

- Thread safety by providing immutable classes
- Domain driven design
  - Different use cases for date and time
- Separation of chronologies
  - API allows people to work with different calendaring systems like Japan / Thailand etc.

Java.time API

Instant	Represents an instant in time on the time line. In the Java 7 date time API an instant was typically represented by a number of millseconds since Jan. 1st. 1970. In Java 8 the Instant class represents an instant in time represented by a number of seconds and a number of nanoseconds since Jan. 1st 1970.
Duration	Represents a duration of time, for instance the time between two instants. Like the Instant class a Duration represents its time as a number of seconds and nanoseconds.
LocalDate	Represents a date without time zone information - e.g. a birthday, official holiday etc.
LocalDateTime	Represents a date and time without time zone information
LocalTime	Represents a local time of day without time zone information.
ZonedDateTime	Represents a date and time including time zone information
Period	Represents duration of time in terms of years, months and days.
DateTimeFormatter	Formats date time objects as Strings. For instance a ZonedDateTime or a LocalDateTime.

Creating and manipulating Dates

```
LocalDate localDate=LocalDate.now();
System.out.println(localDate);

// getting current Time
LocalTime localTime=LocalTime.now();
System.out.println(localTime);

//local dateTime
LocalDateTime dateTime=LocalDateTime.now();

// creating zonedDateTime object
ZonedDateTime zonedDateTime
=ZonedDateTime.of(dateTime,ZoneId.systemDefault());
```

Finding difference between two dates using ChronoUnit

```
LocalDateTime ldtStart = LocalDateTime.of(2015, 10, 23, 12, 7, 1);

LocalDateTime ldtEnd   = LocalDateTime.of(2015, 11, 25, 15, 8, 2);

long numberOfMonths =ChronoUnit.MONTHS.between(ldtStart, ldtEnd);

System.out.println("Between in Months : " + numberOfMonths);

long numberOfDays =ChronoUnit.DAYS.between(ldtStart, ldtEnd);
```

Finding difference between two dates using ChronoUnit

```
System.out.println("Between in Days : " +
numberOfDays);

long numberOfHours =
ChronoUnit.HOURS.between(ldtStart, ldtEnd);

System.out.println("Between in hours : " +
numberOfHours);

long numberOfMinutes =
ChronoUnit.MINUTES.between(ldtStart, ldtEnd);

System.out.println("Between in minutes : " +
numberOfMinutes);
```

Persistent

Finding difference between two dates using ChronoUnit

```
//Using Period class to get the difference between to dates in year, month and days

Period years=Period.between(today, hireDate);

System.out.println(" Completed :"+years.getYears()+" years "+years.getMonths()+" months"+years.getDays()+" days");
```

Formatting dates

```
LocalDate date = LocalDate.now();

DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy MM dd");

// converting date to String
String text = date.format(formatter);

// converting String to dates
LocalDate parsedDate = LocalDate.parse(text, formatter);
```

Summary

With this we have come to an end of our session, where we discussed about

- LocalDate, LocalDateTime, ZonedDateTime class
- To find the difference in two dates using Period class
- Formatting dates using DateTimeFormatter

Appendix

Reference Material : Websites & Blogs

https://docs.oracle.com/javase/8/docs/api/?java/util/function/package-summary.html

https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html

Reference Material: Books

Java SE 8 for the Really Impatient

By: Cay S Horstmann

Java 8 Lambaddas

By: Richard Warburton

Persistent

Thank you!

Persistent Interactive | Persistent University